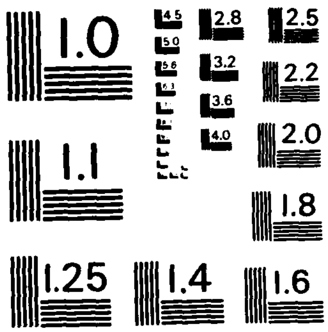
AD-A160 121 DEVELOPMENT OF AN ACCURATE ALGORITHM FOR EXP(BT) 1/1
APPENDIX(U) CALIFORNIA UNIV BERKELEY CENTER FOR PURE
AND APPLIED MATHEMATICS B N PARLETT ET AL. AUG 85

UNCLASSIFIED PAM-294-APP F/G 9/2 NL

END

FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

CENTER FOR PURE AND APPLIED MATHEMATICS
UNIVERSITY OF CALIFORNIA, BERKELEY

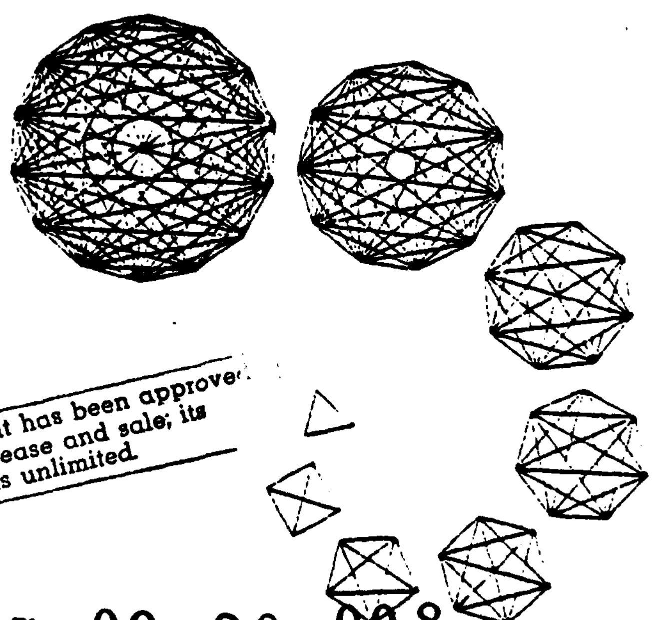
PAM-294

DEVELOPMENT OF AN ACCURATE ALGORITHM FOR EXP(Bτ)

(APPENDIX)

B.N. PARLETT & K.C. NG

AUGUST 1985



This report was done with support from the Center for Pure and Applied Mathematics. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of the Center for Pure and Applied Mathematics or the Department of Mathematics.

2

Appendix: Listing and subroutines details

One sample program (with its output) and 18 subroutines are listed in this appendix.

- sample program and its output
- subroutine cschur
- subroutine cisol
- subroutine cmhu
- subroutine chqr
- subroutine corder
- subroutine cexphy
- subroutine cexpri
- subroutine cindex (in cexpri)
- subroutine cmswap (in cexpri)
- subroutine cexpnt
- subroutine cddexp
- subroutine cfmulv
- complex function cdotc (linpack BLAS, in blas_c)
- complex function cdotu (linpack BLAS, in blas_c)
- subroutine caxpy (linpack BLAS, in blas_c)
- subroutine ccopy (linpack BLAS, in blas_c)
- subroutine csscal (linpack BLAS, in blas_c)
- subroutine cscal (linpack BLAS, in blas_c)

Sample program

```
C
C THIS PROGRAM SHOWS THE USAGE OF SUBROUTINES FOR COMPUTING THE MATRIX
C EXPONENTIAL.
C
      PARAMETER (M=6)
      COMPLEX P(M,M), B(M,M),E(M,M),W(5*M),TAU,V(M,M)
      REAL OVFT
      INTEGER M,N,I,J,IP(M),IDX(M),IERR,NP,NB,NE,NV
C A NUMBER NEAR THE OVERFLOW THRESHOLD ON A VAX IN SINGLE PRECISION
      OVFT=1.0E36
C ROW DIMENSION OF THE MATRICES
      NB=M
      NP=M
      NE=M
      NV=M
C READ IN THE DIMENSION, TAU, AND THE WHOLE MATRIX (BY ROWS)
      READ*, N, TAU, ((B(I,J),J=1,N),I=1,N)
      PRINT*, 'INPUT MATRIX', 'TAU=', TAU
      CALL MATPT(M,N,B)
C SET V=I
      DO 20 J=1,N
      DO 10 I=1,N
10    V(I,J)=CMPLX(0)
20    V(J,J)=CMPLX(1)
C COMPUTE THE MATRIX EXPONENTIAL
      CALL CSCHUR (B,P,W,N,NB,NP,IERR)
      CALL CORDER(B,P,TAU,W,N,NB,NP)
      PRINT*, ' '
      PRINT*, 'SCHUR FORM S'
      CALL MATPT(M,N,B)
      PRINT*, ' '
      PRINT*, 'UNITARY MATRIX P '
      CALL MATPT(M,N,P)
      CALL CEXPHY(B,E,W,IP,IDX,TAU,OVFT,IERR,N,NB,NE)
      IF( IERR.EQ. -2) THEN
          PRINT*, 'THE EXPONENTIAL OF SOME EIGENVALUE OVERFLOWS'
          STOP
      ENDIF
      CALL CFMULV(E,P,V,W,N,NE,NP,NV,N)
      PRINT*, ' '
      PRINT*, 'EXP(TAU*B)'
      CALL MATPT(M,N,V)
      END

      SUBROUTINE MATPT(M,N,X)
      COMPLEX X(M,N)
      DO 10 I=1,N
      WRITE(6,100) (X(I,J),J=1,N)
10    CONTINUE
100   FORMAT(1X,3('(',2G12.3,')'))
      RETURN
      END

INPUT DATA :
6,(1,0)
(1,0)   (-50,0) (0,0)   (1,0)   (1,0)   (1,0)
(50,0)  (1,0)   (0,0)   (1,0)   (1,0)   (1,0)
(0,0)   (0,0)   (1,0)   (100,0) (0,0)   (0,0)
(0,0)   (0,0)   (0,0)   (1,0)   (0,0)   (0,0)
(0,0)   (0,0)   (0,0)   (0,0)   (1,0)   (50,0)
(0,0)   (0,0)   (0,0)   (0,0)   (-50,0) (1,0)
```

We ran the program on a Vax-780 and obtained the following results. For simplicity, only three digits are displayed, and the output has been reformatted so that it is easy to read.

INPUT MATRIX B

1	-50	0	1	1	1
50	1	0	1	1	1
0	0	1	100	0	0
0	0	0	1	0	0
0	0	0	0	1	50
0	0	0	0	-50	1

SCHUR FORM S (AFTER SUBROUTINE *RORDER*)

1-50i	0	-i	i	0	.707+ .707i
0	1+ 50i	i	-i	0	-.707-.707i
0	0	1-50i	0	0	0
0	0	0	1+ 50i	0	0
0	0	0	0	1	100
0	0	0	0	0	1

UNITARY TRANSFORMATION P

-.707i	-.707	0	0	0	0
.707	.707i	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	-.707i	.707	0	0
0	0	-.707	.707i	0	0

EXP(B)

2.62	.713	0	-.0162	.699	2.62
-.713	2.62	0	-.0124	2.62	-.727
0	0	2.72	272.	0	0
0	0	0	2.72	0	0
0	0	0	0	2.62	-.713
0	0	0	0	.713	2.62

CSCHUR

1. PURPOSE

The Fortran 77 subroutine CSCHUR reduces a complex general matrix B to a complex upper Triangular matrix S using unitary similarity transformations.

2. USAGE

(A). Calling Sequence.

```
SUBROUTINE  CSCHUR(B,P,w,n,nb,np,ierr)
```

Parameters :

B — is a complex two-dimensional variable with row dimension nb and column dimension at least n. On input, B contains the complex matrix of order n to be reduced to triangular form. On output, B is overwritten by its Schur form S (upper triangular).

P — is a complex two-dimensional output variable with row dimension np and column dimension at least n. On output, P contains the unitary matrix that transform B to S : $S=P^{*}\cdot B\cdot P$.

w — is a complex one-dimensional variable with dimension 2n. Work space.

n — is an input integer equal to the order of the matrix B.

nb,np — are input integers equal to the row dimension of B and P respectively.

ierr — is an integer variable indicating the error condition. If the number of QR-iterations for some eigenvalue reaches 30 without converging (at that point the subroutine CHQR terminates), then ierr is set equal to -1; otherwise ierr=0.

Subprograms:

CISOL, CMHU, CHQR.

3. DISCUSSION OF METHOD AND ALGORITHM

We first call CISOL to isolate the eigenvalues of B. Then we employ Householder transformations to reduce B to upper Hessenberg form H (subroutine CMHU). Finally we apply QR-algorithm to transform H to upper triangular form (subroutine CHQR). For details of the algorithm, see the description of subroutines CISOL, CMHU and CHQR.

There are approximately 3 executable Fortran statements.

4. REFERENCES

See CISOL, CMHU, and CHQR.

5. PROGRAM LISTING.

```
      SUBROUTINE  CSCHUR (B,P,W,N,NB,NP,IERR)
      COMPLEX B(NB,N),P(NP,N),W(2*N)
      INTEGER N,NB,NP,IERR
C
C CSHCUR FIRST ISOLATES THE EIGENVALUES OF B WHENEVER POSSIBLE, THEN USES
C HOUSEHOLDER TRANSFORMATION TO REDUCE B TO UPPER HESSENBERG H, FINALLY
C USES QR TO TRANSFORM H TO ITS SCHUR FORM S THAT WILL REPLACE B  THE
C TRANSFORMATIONS ARE ACCUMULATED IN P, I.E., S=PH*B*P, HERE PH STANDS
C FOR THE COMPLEX CONJUGATE TRANSPOSE OF P. IF QR FAILS TO CONVERGE, WE
C RETURN IERR=-1, OTHERWISE IERR=0
C CODE IN F77 BY K.C. NG, UC BERKELEY, REVISED ON 5/22/85.
C
C     REQUIRED SUBROUTINES      : CISOL, CMHU, CHQR
C
C GLOSSARY.
C     B      INPUT: A COMPLEX MATRIX; OUTPUT: TRIANGULAR MATRIX
C     P      A UNITARY MATRIX
C     W      COMPLEX ARRAY, WORK SPACE
C -   N      DIMENSION OF MATRIX H
C     NB,NP  LEADING DIMENSION OF ARRAYs B AND P RESPECTIVELY
C     IERR   OUTPUT (INTEGER) ERROR STATUS: IF QR DOESN'T CONVERGE IN 30
C            ITERATION,  THE PROGRAM QUITS AND RETURNS IERR=-1.
C
C INTERNAL VARIABLES
C
      INTEGER LOW,IGH
C
C ISOLATE THE EIGENVALUES IN B
C
      CALL CISOL(B,P,LOW,IGH,N,NB,NP)
C
C TRANSFORM B TO UPPER HESSENBERG FORM
C
      CALL CMHU(B,P,LOW,IGH,N,NB,NP)
C
C TRANSFORM THE UPPER HESSENBERG MATRIX TO TRIANGULAR FORM AND ACCUMULATE
C
      CALL CHQR(B,P,LOW,IGH,W,W(N+1),N,NB,NP,IERR)
      RETURN
      END
```

CISOL
(Subprogram of CSCHUR)

1. PURPOSE

The Fortran 77 subroutine CISOL isolates (using permutations) the eigenvalues of B whenever possible.

2. USAGE

(A). Calling Sequence.

```
SUBROUTINE CISOL(B,P,low,igh,n,nb,np)
```

Parameters :

- **B** is a two-dimensional complex variable with row dimension nb and column dimension at least n. On input, B contains a complex matrix of order n. On output, B contains the permuted matrix.
- **P** is an output two-dimensional complex variable with row dimension nb and column dimension at least n. P contains the permutations.
- **low,igh** are integer output variables indicating the boundary indices for the permuted matrix.
- **n** is an input integer equal to the column dimension of the matrix B.
- **nb,np** are input integers equal to the row dimension of B and P respectively.

3. DISCUSSION OF METHOD AND ALGORITHM

This is a F77 variation of the first half (isolating the eigenvalues of B whenever possible) of *cbal*, an EISPACK Fortran IV subroutine. The permutations are recorded in a matrix P. For details of the algorithm, see EISPACK [2] or Parlett and Reinsch [1].

There are approximately 42 executable Fortran statements.

4. REFERENCES

[1] B.N. Parlett and C. Reinsch, *Balancing a Matrix for Calculation of Eigenvalues and Eigenvectors*, Num. Math. 13, 293-304 (1969). (Reprinted in Handbook for Automatic Computation, Vol. II, Linear Algebra, J.H. Wilkinson - C, Reinsch, Contribution II/11, 315-326, Springer-Verlag, 1971.)

[2] B.T. Smith & et al, *Matrix Eigensystem Routines - EISPACK Guide*, Lecture Notes in Computer Science, Vol 6, Springer-Verlag, 1974.

5. PROGRAM LISTING.

```
      SUBROUTINE CISOL(B,P,LOW,IGH,N,NB,NP)
      COMPLEX B(NB,N),P(NP,N)
      INTEGER NB,N,LOW,IGH,NP
C
C CISOL ISOLATES THE EIGENVALUES IN THE INPUT COMPLEX MATRIX B. IT IS A F7
C VARIATION OF THE FIRST HALF OF EISPACK SUBROUTINE CBAL, WITH THE RESULTED
C PERMUTATION RECORDED IN P   P WILL BE SET TO AN IDENTITY MATRIX INITIALY
C CODE IN F77 BY K.C. NG, UC BERKELEY; REVISED ON 5/22/85.
C
C GLOSSARY:
C      B       A COMPLEX MATRIX
C      P       ON OUTPUT, AN UNITARY MATRIX
C      IGH,LOW        INTEGER INDICES INDICATING BOUNDARIES
C      N       DIMENSION OF MATRIX B
C      NB,NP  LEADING DIMENSION OF ARRAYS B AND P RESPECTIVELY
C
C
C INTERNAL VARIABLES
C
      INTEGER I,J,K,L,M,IEXC
      COMPLEX X,ZERO
      DATA ZERO/(0E0,0E0)/
C
C SET P=I
C
      DO 10 J=1,N
      DO 5 I=1,N
5     P(I,J)=ZERO
10    P(J,J)=1.0
C
      K=1
      L=N
      GOTO 100
C
C IN-LINE PROCEDURE FOR ROW AND COLUMN EXCHANGE
C
20    CONTINUE
      IF(J.NE.M) THEN
          DO 30 I=1,L
            X=B(I,J)
            B(I,J)=B(I,M)
            B(I,M)=X
30        CONTINUE
          DO 40 I=K,N
            X=B(J,I)
            B(J,I)=B(M,I)
            B(M,I)=X
40        CONTINUE
          DO 60 I=1,N
            X=P(I,J)
            P(I,J)=P(I,M)
            P(I,M)=X
60        CONTINUE
      ENDIF
      IF(IEXC.EQ.2) GOTO 130
C
C SEARCH FOR ROWS ISOLATING AN EIGENVALUE AND PUSH THEM DOWN
C
      IF(L.EQ.1) GOTO 280
      L=L-1
100   DO 120 J=L,1,-1
```

```
            DO 110 I=1,L
              IF(I.EQ.J) GOTO 110
              IF(B(J,I).NE.ZERO) GOTO 120
110         CONTINUE
            M=L
            IEXC=1
            GOTO 20
120      CONTINUE
         GOTO 140
C
C SEARCH FOR COLUMNS ISOLATING AN EIGENVALUE AND PUSH THEM LEFT
C
130      K=K+1
140      DO 170 J=K,L
            DO 150 I=K,L
              IF(I.EQ.J) GOTO 150
              IF(B(I,J).NE.ZERO) GOTO 170
150         CONTINUE
            M=K
            IEXC=2
            GOTO 20
170      CONTINUE
280      LOW=K
         IGH=L
         RETURN
         END
```

CMHU
(Subprogram of CSCHUR)

1. PURPOSE

The Fortran 77 subroutine CMHU reduces a complex general matrix B to a complex upper Hessenberg matrix H using unitary similarity transformations. The Hessenberg matrix H is needed in subroutine CHQR to find the Schur form of B.

2. USAGE

(A). Calling Sequence.

```
SUBROUTINE  CMHU(B,P,low,igh,n,nb,np)
```

Parameters :

B — is a complex two-dimensional variable with row dimension nb and column dimension at least n. On input, B contains the complex matrix of order n to be reduced to Hessenberg form. On output, B is overwritten by the upper Hessenberg matrix H.

P — is a complex two-dimensional variable with row dimension np and column dimension at least n. On output, P contains the unitary matrix that transform B to H : $H = P^{*} \cdot B \cdot P$.

low,igh — are integer input variables indicating the boundary indices for the matrix. See CISOL for details.

n — is a input integer equal to the order of the matrix B.

nb,np — are input integers equal to the row dimension of B and P respectively.

3. DISCUSSION OF METHOD AND ALGORITHM

The method employs Householder transformations to reduce B to upper Hessenberg form. A general description (for complex B) can be found in [1], pp.137-138. For real matrices, EISPACK [3] (see also [4]) has a subroutine *orthes* to do the reduction. The subroutine CMHU given here is essentially an extension of *orthes* to complex matrices. (EISPACK has another subroutine *elmhes* to reduce a general complex matrix to Hessenberg form, but the transformations are non-unitary.)

The details of the method can be illustrated by a 6 by 6 example. A typical stage in the reduction would be

$$B = \begin{bmatrix} h & h & h & t & t & t \\ h & h & h & t & t & t \\ 0 & h & h & t & t & t \\ 0 & 0 & x1 & f & f & f \\ 0 & 0 & x2 & f & f & f \\ 0 & 0 & x3 & f & f & f \end{bmatrix} = \left[\begin{array}{cc|c} \multicolumn{2}{c|}{H} & T \\ \hline 0 & \multicolumn{1}{|c|}{x} & F \end{array}\right]$$

where the elements h are those of the final Hessenberg matrix, and the x's , t and f are in their intermediate states. Let T and F denote the 3 by 3 sub-matrices of B consisting of small letters t and f respectively. Also let x denote the vector $(x1,x2,x3)^t$, as shown in the above figure. The first step of the Householder transformation is to find a vector u to form the Householder matrix R such that

$$R \cdot \begin{pmatrix} x1 \\ x2 \\ x3 \end{pmatrix} = \begin{pmatrix} w \\ 0 \\ 0 \end{pmatrix}, \quad \text{where } R := I - \frac{uu^H}{d,} \quad d = u^H u/2\,;$$

then the sub-matrices F and T of B are updated, and the transformation is accumulated in P as indicated :

$$F := R \cdot F \cdot R,$$
$$T := T \cdot R,$$
$$P := P \cdot \begin{pmatrix} I & \\ & R \end{pmatrix}.$$

The determination of w and the vector u is as follows. For $x1 \neq 0$, it is easy to show that

$$w = \pm \frac{x1}{|x1|}\sqrt{S}, \quad S := |x1|^2 + |x2|^2 + |x3|^2.$$

When $x1=0$, w can be any complex number with magnitude $\sqrt{S}$. In CMHU, we choose w to be $(x1/|x1|)\sqrt{S}$ if $x1 \neq 0$ and $\sqrt{S}$ if $x1=0$. The vector u is just equal to $(x1-w,x2,x3)^t$. However, care must be taken in computing the first component $u_1 = x1-w$ when x1 and w are almost equal (cf. [2], p.91). The following formula is used whenever $|w/x1|<2$:

$$u_1 = x1 - w = \frac{-(|x2|^2 + |x3|^2)}{(\overline{x1} + \overline{w})}.$$

The above steps are repeated on further columns of the transformed B until it reaches the Hessenberg form.

There are approximately 45 executable Fortran statements.

4. REFERENCES

[1] C.E. Fröberg, *Introduction to Numerical Analysis,* 2nd ed,. Addison- Wesley 1969.

[2] B.N. Parlett, *The Symmetric Eigenvalue Problem,* Prentice-Hall, Englewood

Cliffs, N.J. 1980.

[3] B.T. Smith & et al, *Matrix Eigensystem Routines - EISPACK Guide*, Lecture Notes in Computer Science, Vol 6, Springer-Verlag, 1974.

[4] J.H. Wilkinson & C. Reinsch, Handbook in Automatic Computation Vol. II, *Linear Algebra in Numerical Analysis*, Springer-Verlag, 1971.

5. PROGRAM LISTING.

```
      SUBROUTINE  CMHU(B,P,LOW,IGH,N,NB,NP)
      COMPLEX B(NB,N),  P(NP,N)
      INTEGER LOW,IGH,N,NB,NP
C
C THIS SUBROUTINE REDUCES A COMPLEX MATRIX B TO UPPER HESSENBERG FORM BY
C UNITARY TRANSFORMATIONS. TRANSFORMATIONS ARE ACCUMULATED IN P. THIS
C SUBROUTINE SHOULD BE PRECEEDED BY CISOL.
C CODE IN F77 BY K.C. NG, UC BERKELEY; REVISED ON 5/22/85.
C
C      F77  GENERIC FUNCTIONS     :  ABS ,CONJG, IMAG, REAL, SQRT
C      F77  NON-GENERIC FUNCTION:  CMPLX
C
C      REQUIRED BLAS SUBROUTINES:  CAXPY, CCOPY, CSCAL, CSSCAL
C      REQUIRED BLAS FUNCTION    :  CDOTC
C
C GLOSSARY:
C      B       A COMPLEX MATRIX
C      P       A UNITARY MATRIX
C      IGH,LOW       INTEGER INDICES INDICATING BOUNDARIES
C      N       DIMENSION OF MATRIX B, P
C      NB,NP LEADING DIMENSION OF ARRAYs B AND P RESPECTIVELY
C
C
C
C INTERNAL VARIABLES
C
      COMPLEX X,Y,Z,ZERO,CDOTC
      REAL T,D,HALF,SCALE
      INTEGER I,J,R,RP1
C
      ZERO=0
      HALF=0.5
      IF (IGH-LOW.LT.2) RETURN
      DO 200 R=LOW,IGH-2
C
C SCALE COLUMN AND SET UP THE HOUSEHOLDER VECTOR IN B(.,R)
C
         RP1=R+1
         SCALE=0
         DO 20 I=RP1,IGH
20       SCALE=SCALE+ABS(REAL(B(I,R)))+ABS(IMAG(B(I,R)))
         IF(SCALE.EQ.REAL(ZERO)) RETURN
         Y=0
         DO 50 I=R+2,IGH
         B(I,R)=B(I,R)/SCALE
50       Y=Y+B(I,R)*CONJG(B(I,R))
         Y=CMPLX(REAL(Y))
         IF(Y.EQ.ZERO) GOTO 200
         X=B(RP1,R)/SCALE
         IF(X.EQ.ZERO) THEN
           D=REAL(Y)
           Z=CMPLX(SQRT(REAL(Y)))
           B(RP1,R)=-Z
         ELSE
           T=SQRT(REAL(Y+X*CONJG(X)))/ABS(X)
           Z=X*CMPLX(T)
           IF(T.LT.HALF) THEN
                B(RP1,R)=X-Z
           ELSE
C
C CANCELLATION OCCURS, USE RATIONALIZED FORMULA FOR X-Z
```

```
C
                  B(RP1,R)=-Y/CONJG(X+Z)
              ENDIF
              D=REAL(Y+B(RP1,R)*CONJG(B(RP1,R)))*HALF
           ENDIF
C
C PERFORM B-((B*U)/D)*UT, P-((P*U)/D)*UT, UT IS THE CONJUGATE TRANSPOSE OF
C
           DO 90 I=1,N
              X=0
              Y=0
              DO 70 J=RP1,IGH
                  X=X+B(I,J)*B(J,R)
                  Y=Y+P(I,J)*B(J,R)
70            CONTINUE
              X=X/D
              Y=Y/D
              DO 80 J=RP1,IGH
                  B(I,J)=B(I,J)-CONJG(B(J,R))*X
                  P(I,J)=P(I,J)-CONJG(B(J,R))*Y
80            CONTINUE
90         CONTINUE
C
C PERFORM  B-U*((UH*B)/D)
C
           DO 120 J=RP1,N
              X=CDOTC(IGH-RP1+1,B(RP1,R),1,B(RP1,J),1)
              X=X/D
              CALL CAXPY(IGH-RP1+1,-X,B(RP1,R),1,B(RP1,J),1)
120        CONTINUE
           B(RP1,R)=Z*SCALE
C
C CLEAR THE ELEMENTS BELOW THE SUB-DIAGONAL
C
           DO 130 I=R+2,IGH
130        B(I,R)=0
200     CONTINUE
        RETURN
        END
```

CHQR
(Subprogram of CSCHUR)

1. PURPOSE

The Fortran 77 subroutine CHQR reduces a complex upper Hessenberg matrix H to a complex upper triangular matrix S (Schur form) using the QR-algorithm. The unitary transformations are accumulated in P.

2. USAGE

(A). Calling Sequence.

SUBROUTINE CHQR(H,P,low,igh,w,v,n,nh,np,ierr)

Parameters:

H — is a complex two-dimensional variable with row dimension nh and column dimension at least n. On input, it contains the complex upper Hessenberg matrix H. On output, it contains the triangular matrix S, the Schur form of H.

P — is a complex two-dimensional variable with row dimension np and column dimension at least n. On input, P contains an unitary matrix that changed B to H : $H=P^{*}\cdot B\cdot P$ (obtained from subroutine CMHU). On output, P contains the unitary matrix that transforms B to its Schur form S : $S=P^{*}\cdot B\cdot P$.

low,igh — are integer input variables indicating the boundary indices for the matrix. See subroutine CISOL for details.

w,v — are two complex one-dimensional variables of order n. Work space.

n — is an input integer equal to the order of the matrix H.

nh,np — are input integers equal to the row dimension of H and P respectively.

ierr — is an integer variable indicating the error condition. If the number of QR-iterations for some eigenvalue reaches 30 without getting convergent (the subroutine terminates if that happens), then ierr is set equal to -1; otherwise ierr = 0.

3. DISCUSSION OF METHOD AND ALGORITHM

CHQR is essentially a complex version of the EISPACK [3,4] subroutine *hqr* (or *hqr2*) with some modifications. The method is to apply the QR-algorithm [1] to the Hessenberg matrix H to reduce it to triangular form. Since the QR-algorithm is a popular topic in many numerical analysis texts (eg. [2],[4]), we will not repeat the general description here. Rather, we discuss certain technical details which are different from

EISPACK's *hqr* in the implementation of the QR-algorithm. There are four major differences.

(1). First of all, complex arithmetic is used throughout CHQR; therefore, an orthogonal matrix becomes an unitary matrix as described in [1] and double shifting for conjugate eigenvalues is no longer needed.

(2). The criterion for testing a negligible sub-diagonal is as follows. Let $|x|$ stand for $|Re(x)| + |Im(x)|$, and let ϵ denote the machine precision. Then H(i,i-1) is negligible if $|H(i,i-1)|$
satisfies (a) and (b1), or (a) and (b2) if (b1) fails:

(a). $|H(i,i-1)| \leq \epsilon \cdot \min\{|H(i,i)|, |H(i-1,i-1)|\}$;
(b1). $4|H(i,i-1)*H(i-1,i)| \leq \epsilon \cdot (|H(i,i)-H(i-1,i-1)|^2)$;
(b2). $2\sqrt{|H(i,i)*H(i-1,i-1)|} \leq \epsilon \cdot |H(i,i) + H(i-1,i-1)|$.

These criteria are obtained by forcing the eigenvalues of the following 2×2 sub-matrix to agree in working precision with the eigenvalues of the same matrix with H(i,i-1) replaced by zero.

$$\begin{bmatrix} H(i-1,i-1) & H(i-1,i) \\ H(i,i-1) & H(i,i) \end{bmatrix}.$$

(3). CHQR doesn't look for two consecutive small sub-diagonal elements.

(4). The transformations are accumulated in P.

There are approximately 70 executable Fortran statements.

4. REFERENCES

[1] J.G.F. Francis, *The QR Transformation, Part I & II,* Computer Journal 4 (1961/62), pp. 265-271, 332-345.

[2] G.H. Golub & C.F. VanLoan, *Matrix Computation,* Johns Hopkins University Press, 1983.

[3] B.T. Smith & et al, *Matrix Eigensystem Routines - EISPACK Guide,* Lecture Notes in Computer Science, Vol 6, Springer-Verlag, 1974.

[4] J.H. Wilkinson, *The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.*

[5] J.H. Wilkinson & C. Reinsch, Handbook in Automatic Computation Volume II, *Linear Algebra in Numerical Analysis,* Springer-Verlag, Berlin, Heidelberg, New York, 1971.

5. PROGRAM LISTING.

```
      SUBROUTINE  CHQR (H,P,LOW,IGH,W,V,N,NH,NP,IERR)
      COMPLEX H(NH,N),P(NP,N),W(N),V(N)
      INTEGER LOW,IGH,N,NH,NP,IERR
C
C THIS SUBROUTINE USES QR-ALGORITHM TO TRANSFORM A COMPLEX UPPER HESSENBEB
C MATRIX H TO ITS SCHUR FORM. THE TRANSFORMATIONS ARE ACCUMULATED IN P. THE
C SUBROUTINE FAILS (RETURN IERR = -1 ) IF ANY EIGENVALUE TAKES MORE THAN 8
C ITERATIONS.
C CODE IN F77 BY K.C. NG, UC BERKELEY; REVISED ON 5/22/85.
C
C     F77  GENERIC FUNCTIONS     :  ABS, CONJG, IMAG, MAX, REAL, SQRT
C     F77  NON-GENERIC FUNCTION:  CMPLX
C
C GLOSSARY:
C     H       INPUT: A COMPLEX HESSENBERG MATRIX; OUTPUT: TRIANGULAR MATRIX
C     P       AN UNITARY MATRIX
C     W,V     COMPLEX ARRAIES, WORK SPACE
C     IGH,LOW       INTEGER INDICES INDICATING BOUNDARIES
C     N       DIMENSION OF MATRIX H
C     NH,NP LEADING DIMENSION OF ARRAIES H AND P RESPECTIVELY
C     IERR  OUTPUT (INTEGER) ERROR STATUS: IF QR DOESN'T CONVERGE IN 30
C             ITERATION,  THE PROGRAM QUITS AND RETURNS IERR=-1.
C
C INTERNAL VARIABLES
C
      COMPLEX X,Y,Z,EI1,EI2,SHIFT,ZERO
      REAL T1,T2,T3,AB,HMAX,HALF
      INTEGER I,J,K,M,ITS
C
C DEFINE AB(X)  := |REX|+|IMX|
C
      AB(X)=ABS(REAL(X))+ABS(IMAG(X))
      ZERO=CMPLX(0.0)
      HALF=0.5
      M=IGH
      ITS=0
10    IF(M.LE.LOW) GOTO 300
      DO 100 K=M,LOW+1,-1
C
C LOOK FOR SINGLE SMALL SUB-DAIGONAL ELEMENT
C
          T1=MIN(AB(H(K-1,K-1)),AB(H(K,K)))
          T3=AB(H(K,K-1))
          T2=T1+T3
          IF(T1.EQ.T2) THEN
            HMAX=MAX(T3,AB(H(K-1,K)),AB(H(K-1,K-1)),AB(H(K,K)))
            T1=(AB(H(K-1,K-1)-H(K,K))/HMAX)**2
            T3=4*(T3/HMAX)*(AB(H(K-1,K))/HMAX)
            T2=T1+T3
            IF(T1.EQ.T2) GOTO 110
            T1=AB(H(K-1,K-1)+H(K,K))/HMAX
            T2=T1+SQRT(T3)
            IF(T1.EQ.T2) GOTO 110
          ENDIF
100   CONTINUE
110   IF(K.EQ.M) THEN
C
C ONE EIGENVALUE FOUND
C
          M=M-1
          ITS=0
```

```
          GOTO 10
       ENDIF
       IF(ITS.EQ.30) THEN
          IERR=-1
          GOTO 300
       ENDIF
       IF(ITS.EQ.10.OR.ITS.EQ.20) THEN
C
C FORM EXCEPTIONAL SHIFT
C
          SHIFT=H(M,M-1)
       ELSE
C
C FORM WILKINSON'S SHIFT
C
          Z=H(M,M-1)*H(M-1,M)
          X=SQRT((H(M-1,M-1)-H(M,M))**2+4.0*Z)
          Y=H(M-1,M-1)+H(M,M)
          EI1=(X+Y)*HALF
          IF(AB(X+Y).LT.AB(Y-X)) EI1=(Y-X)*HALF
          EI2=Y-EI1
          IF(4*AB(EI2).LT.AB(Y)) EI2=(H(M,M)*H(M-1,M-1)-Z)/EI1
          SHIFT=EI1
          IF(AB(EI1-H(M,M)).GT.AB(EI2-H(M,M))) SHIFT=EI2
       ENDIF
       ITS=ITS+1
       DO 120 I=K,M
120    H(I,I)=H(I,I)-SHIFT
C
C QR ITERATION:   ROW MODIFICATION
C
       DO 140 I=K,M-1
          T1=SQRT(REAL(H(I,I)*CONJG(H(I,I)))+
     X       REAL(H(I+1,I)*CONJG(H(I+1,I))))
          W(I)=H(I,I)/T1
          V(I)=H(I+1,I)/T1
          H(I,I)=CMPLX(T1)
          H(I+1,I)=ZERO
          DO 130 J=I+1,N
             X=H(I,J)
             H(I,J)=CONJG(W(I))*X+CONJG(V(I))*H(I+1,J)
             H(I+1,J)=W(I)*H(I+1,J)-V(I)*X
130       CONTINUE
140    CONTINUE
C
C               COLUMN MODIFICATION
C
       DO 180 J=K,M-1
          DO 160 I=1,J
             X=H(I,J)
             H(I,J)=W(J)*X+V(J)*H(I,J+1)
             H(I,J+1)=CONJG(W(J))*H(I,J+1)-CONJG(V(J))*X
160       CONTINUE
          H(J+1,J)=V(J)*H(J+1,J+1)
          H(J+1,J+1)=CONJG(W(J))*H(J+1,J+1)
C
C             ACCUMULATE TRANSFORMATION
C
          DO 170 I=1,N
             X=P(I,J)
             P(I,J)=W(J)*X+V(J)*P(I,J+1)
             P(I,J+1)=CONJG(W(J))*P(I,J+1)-CONJG(V(J))*X
170       CONTINUE
180    CONTINUE
```

```
C
C           SHIFT BACK
C
      DO 210 I=K,M
210   H(I,I)=H(I,I)+SHIFT
      GOTO 10
C
C CLEAR THE LOWER TRIANGULAR PART
C
300   DO 350 J=1,N-1
      DO 350 I=J+1,N
350   H(I,J)=ZERO
      END
```

CORDER

1. PURPOSE.

The Fortran 77 subroutine CORDER re-orders the diagonal of a complex triangular matrix S according to the real parts of tau*S(i,i), using unitary similarity transformations that preserves triangularity. The transformations are accumulated in P.

2. USAGE.

(A). Calling Sequence.

```
SUBROUTINE CORDER(S,P,tau,w,n,ns,np)
```

Parameters :

S is a complex two-dimensional variable with row dimension ns and column dimension at least n. On input, S contains a complex triangular matrix of order n (a Schur form of B). On output, S contains the triangular matrix which is unitarily similar to the previous one and has its diagonal elements ordered according the the real parts of the diagonal of tau*S, where tau is a complex parameter.

P is a complex two-dimensional variable with row dimension np and column dimension at least n. On input, P contains an unitary matrix that transform B to its Schur form S: $S=P^{*}\cdot B\cdot P$. On output, P contains the updated unitary matrix that transform B to the new S.

w is a real one-dimensional variable of order n. Work space.

tau is an input complex number, parameter.

n is an input integer equal to the order of the matrix B.

ns,np are input integers equal to the row dimension of S and P respectively.

3. DISCUSSION OF METHOD AND ALGORITHM.

The re-ordering of the diagonals of S accomplished by a sequence of adjacent swaps. A bubble sort is performed on elements w(i):=Re(tau*S(i,i)). Whenever a swap is needed, say, to swap w(k) and w(k+1), an unitary matrix U is constructed so that

$$U^{*}\cdot\begin{pmatrix} s11 & s12 \\ 0 & s22 \end{pmatrix}\cdot U = \begin{pmatrix} s22 & s12 \\ 0 & s11 \end{pmatrix},$$

here s11, s12 and s22 are the current elements S(k,k), S(k,k+1) and S(k+1,k+1) respectively. It is elementary to show that the U defined below effects such a transformation:

$$
U = \begin{cases} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & \text{if } s11{=}s22\ ; \\[2ex] \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & \text{if } s12{=}0 \text{ and } s11{\neq}s22\ ; \\[2ex] \begin{pmatrix} x & -x/\bar{x} \\ 1 & x \end{pmatrix} \Big/ \sqrt{1+|x|^2}, \quad x := \dfrac{s12}{s22-s11} & \text{otherwise.} \end{cases}
$$

To complete the transformation, columns k and (k+ 1) of both S and P are postmultiplied by U; rows k and (k+ 1) of the new S are premultiplied by U^*.

The algorithm terminates when the bubble sort is finished, or when no swap occurs in any sweep (which means that all the elements are in order).

There are approximately 67 executable Fortran statements.

4. REFERENCES.

None

5. PROGRAM LISTING.

```
      SUBROUTINE CORDER(S,P,TAU,W,N,NS,NP)
      COMPLEX S(NS,N),P(NP,N),TAU
      REAL W(N)
      INTEGER N,NS,NP
C
C THIS SUBROUTINE RE-ORDERS THE DIAGONAL OF A COMPLEX TRIANGULAR MATRIX S
C BY UNITARY TRANSFORMATIONS (KEEPING S TRIANGULAR). THE NEW ORDERING IS
C ACCORDING TO THE REAL PARTS OF TAU*T(I,I). THE TRANSFORMATIONS ARE
C ACCUMULATED IN P.
C CODE IN F77 BY K.C. NG, UC BERKELEY; REVISED ON 5/20/85.
C
C      F77  GENERIC FUNCTIONS     :  ABS, CONJG, REAL, SQRT
C      F77  NON-GENERIC FUNCTION:  CMPLX
C
C GLOSSARY:
C      S       AN UPPER TRIANGULAR COMPLEX MATRIX WHOSE DIAGONALS ARE GOING
C              TO BE RE-ORDERED
C      P       A UNITARY MATRIX IN WHICH THE TRANSFORMATIONS ARE ACCUMULATED
C      TAU     COMPLEX PARAMETER
C      W       COMPLEX ARRAYS, WORK SPACE
C      N       DIMENSION OF MATRIX S,P
C      NS,NP LEADING DIMENSION OF ARRAYS S AND P RESPECTIVELY
C
C INTERNAL VARIABLES
C
      INTEGER I,J,K,NSWAP
      COMPLEX X,Y,U21,U11,U12,ZERO
      REAL RX,RY,SQ,ONE
      ZERO=CMPLX(0.0)
      ONE=1
      DO 10 I=1,N
10    W(I)=REAL(TAU*S(I,I))
      DO 110 I=2,N
      NSWAP=0
      DO 100 K=N,I,-1
      IF (W(K).LT.W(K-1)) THEN
          NSWAP=NSWAP+1
          RX=W(K)
          W(K)=W(K-1)
          W(K-1)=RX
C
C ***** CONSTRUCT THE UNITARY MATRIX U *****
C
          X=S(K-1,K)
          Y=S(K,K)-S(K-1,K-1)
          IF (X.EQ.ZERO) THEN
             U11=0.0
             U21=1.0
             U12=U21
          ELSE
             RX=ABS(X)
             RY=ABS(Y)
             IF (RX.LE.RY) THEN
                  SQ=1.0+(RX/RY)**2
                  IF (SQ.NE.ONE) GOTO 30
C
C      |X| < |Y| AND |X/Y| << 1 (I.E., 1+|X/Y|**2 = 1)
C
                  U11=X/Y
                  U21=1.0
                  U12=-(X/CONJG(X))*(CONJG(Y)/Y)
```

```
              ELSE
                 SQ=1.0+(RY/RX)**2
                 IF (SQ.NE.ONE) GOTO 30
C
C      |X| > |Y| AND  |X/Y| >> 1 (I.E., 1+|X/Y|^2 = |X/Y|^2)
C
                 U11=(X/CMPLX(RX))*(CMPLX(RY)/Y)
                 U21=CMPLX(RY/RX)
                 U12=-U11*CONJG(Y/X)
              ENDIF
           ENDIF
           GOTO 40
C
C      |X/Y| IS NOT TOO BIG AND NOT TOO SMALL
C
30         X=X/Y
           U21=CMPLX(1.0/SQRT(1.0+(RX/RY)**2))
           U11=X*U21
           U12=-U11/CONJG(X)
C
C ***** END OF THE CONSTRUCTION OF U *****
C
C  COLUMN MODIFICATION
C
40         DO 60 J=1,K-2
              X        =S(J,K-1)*U11+S(J,K)*U21
              S(J,K)   =S(J,K-1)*U12+S(J,K)*U11
              S(J,K-1)=X
60         CONTINUE
C
C  ACCUMULATE THE TRANSFORMATION
C
           DO 70 J=1,N
              X        =P(J,K-1)*U11+P(J,K)*U21
              P(J,K)   =P(J,K-1)*U12+P(J,K)*U11
              P(J,K-1)=X
70         CONTINUE
C
C  ROW MODIFICATION
C
           U11=CONJG(U11)
           U21=CONJG(U21)
           U12=CONJG(U12)
           DO 80 J=K+1,N
              X        =S(K-1,J)*U11+S(K,J)*U21
              S(K,J)   =S(K-1,J)*U12+S(K,J)*U11
              S(K-1,J)=X
80         CONTINUE
C
C  SWAP THE DIAGONAL
C
           X      =S(K,K)
           S(K,K)   =S(K-1,K-1)
           S(K-1,K-1)=X
        ENDIF
100     CONTINUE
        IF (NSWAP.EQ.0) RETURN
110     CONTINUE
        RETURN
        END
```

CEXPHY

1. PURPOSE.

The Fortran 77 subroutine CEXPHY computes the exponential of a complex triangular matrix E=exp(tau*B) by first determining a block diagonal clustering on B, calling subroutine CEXPRI to compute the exponential of the diagonal blocks and finally applying Parlett's recurrence formula to fill up the all the off-diagonal elements.

2. USAGE.

Calling Sequence.

SUBROUTINE CEXPHY(B,E,w,ip,idx,tau,ovft,ierr,n,nb,ne)

Parameters :

B	is an input complex two-dimensional variable with row dimension nb and column dimension at least n. B contains a complex triangular matrix of order n, with its diagonal elements ordered according to the real part of tau*B(i,i) (cf. subroutine CORDER).
E	is a complex two-dimensional variable with row dimension ne and column dimension at least n. On output, E contains the exponential of tau*B.
w	is a complex one-dimensional variable with dimension at least 5n. Work space.
ip,idx	are integer arrays of dimension n. Work space.
ovft	is a real number equal to the overflow threshold.
tau	is an input complex number, parameter.
ierr	is an integer variable indicating error condition. If the exponential of some eigenvalues overflow, ierr is set equal to -2, and the subroutine terminates.
n	is an input integer equal to the order of the matrix B.
nb,ne	are input integers equal to the row dimension of B and E respectively.

Subprograms : CEXPRI

3. DISCUSSION OF METHOD AND ALGORITHM.

Let z(1),z(2),...,z(n) denote the eigenvalues of tau*B. CEXPHY first determines a clustering of the diagonals of E=exp(tau*B) (see Fig. 1 in the next page). The requirement is that z(i) in different clusters must be well separated. A typical criterion reads as follows : the i-th and j-th diagonal elements of E are in the same cluster if

$$|z(i)-z(j)| \le g(j-i)\ .$$

We have found that a second degree polynomial is sufficient : $g(k)=a_1+a_2 \cdot k+a_3 \cdot k^2$. (See the program listing for the values of the constants a_i.) This criterion arose from studying the computation of the exponential divided differences. (In [2] it suggests g(k) g(k)=k*(1+ 0.1*ln k); however, we found that a quadratic polynomial is more realistic.)

If two clusters so determined overlap, then we merge them together. The following figure is a typical clustering of E.

$$\begin{bmatrix} X & X & \cdot & \cdot & \cdot & \cdot & \cdot \\ & X & \cdot & \cdot & \cdot & \cdot & \cdot \\ & & X & X & X & \cdot & \cdot \\ & & & X & X & \cdot & \cdot \\ & & & & X & \cdot & \cdot \\ & & & & & X & X \\ & & & & & & X \end{bmatrix} \qquad \begin{array}{ll} X & \text{computed by CEXPNT} \\ \cdot & \text{computed by Parlett's recurrence} \end{array}$$

Fig. 1. Computing exp(tau*B) by hybrid method.

After the clusters are determined, the subroutine CEXPRI is called to compute the diagonal blocks of E (see the description of CEXPRI for details of the method).

Finally, the rest of E is computed by Parlett's recurrence (cf. [3] or [1]) :

$$E(i,j)=\frac{B(i,j)*(E(j,j)-E(i,i))+\sum_{k=i+1}^{j-1}(B(i,k)*E(k,j)-E(i,k)*B(k,j))}{(B(j,j)-B(i,i))}.$$

The recurrence is obtained from the commutativity of E and B. (For real quasi triangular B, the diagonal may contain some 2×2 blocks. In that case, we solve BE-EB=0 directly for each block in turn.)

There are approximately 55 executable Fortran statements.

4. REFERENCES.

[1] G.H. Golub & C.F. VanLoan, *Matrix Computations,* Johns Hopkins University Press, 1983.

[2] A. McCurdy, K.C. Ng and B.N. Parlett, *Accurate Computation of Divided Differences of The Exponential Function,* Mathematics of Computation, Volume 43, Number 168, October 1984, pp501-528.

[3] B.N. Parlett, *A Recurrence Among the Elements of Functions of Triangular Matrices,* Linear Algebra Appl., 14(1976), pp.117-121.

5. PROGRAM LISTING.

```
      SUBROUTINE CEXPHY(B,E,W,IP,IDX,TAU,OVFT,IERR,N,NB,NE)
      COMPLEX B(NB,N), E(NE,N), W(5*N), TAU
      REAL OVFT
      INTEGER N,NB,NE,IERR,IDX(N),IP(N)
C
C THIS SUBROUTINE COMPUTES THE EXPONENTIAL OF (COMPLEX TRIANGULAR) B BY
C CEXPRI AND PARLETT'S RECURRENCE. RESULT STORED IN E
C CODE IN F77 BY K.C. NG, UC BERKELEY, REVISED ON 5/20/85.
C
C     F77  GENERIC FUNCTIONS        ABS, IMAG, REAL, MAX
C
C     REQUIRED SUBROUTINE.       :  CEXPRI
C
C     LOCAL REAL VALUED FUNCTIONS  :  AB, G
C
C GLOSSARY:
C     B      AN UPPER TRIANGULAR COMPLEX MATRIX
C     E      ON OUTPUT, THE EXPONENTIAL OF B
C     W      COMPLEX ARRAYS, WORK SPACE
C     IP,IDX INTEGER ARRAYS, WORK SPACE
C     TAU    COMPLEX PARAMETER
C     OVFT   COMPUTER (REAL) OVERFLOW THRESHOLD
C     IERR   (INTEGER) ERROR STATUS: RETURN IERR= -2 IF SOME EIGENVALUE
C            IS TOO LARGE TO EXPOENETIATE.
C     N      DIMENSION OF MATRIX E
C     NB,NE  LEADING DIMENSION OF ARRAYS B AND E RESPECTIVELY
C
C INTERNAL VARIABLES
C
      COMPLEX Z
      INTEGER I,J,I1,J1,K
      REAL G,A1,A2,A3,AB,ZERO
      DATA A1,A2,A3,ZERO/-1.797804884,1.958414640,.02939024390,0.0/
C
C DEFINE REAL VALUED FUNCTION G AND AB :
C
      G(I)=(A1+I*(A2+I*A3))
      AB(Z)=ABS(REAL(Z))+ABS(IMAG(Z))
C
C INITIALIZE (SAVE THE EIGENVALUES IN THE LAST COLUMN OF E TEMPORARILY)
C
      DO 20 J=1,N-1
      DO 20 I=1,J
20    E(I,J)=B(I,J)*TAU
      DO 40 I=1,N
40    E(I,N)=B(I,I)*TAU
C
C MAKE ROOM IN B
C
      DO 100 J=1,N/2
          K=N-J
          DO 80 I=1,J
80        B(K+I,K)=B(I,J)
100   CONTINUE
C
C FIND THE NEXT CLUSTER (I1,J1)
C
      I1=1
120   J1=I1
      IF(I1.GT.N) GOTO 260
C
```

```
C *** FOR I=I1, I+1 WHILE I < J1+1 DO ****
C
      I=I1
140   IF(I.LE.J1) THEN
          DO 160 J=N,J1,-1
            IF(AB(E(I,N)-E(J,N)).LE G(J-I)) GOTO 180
160       CONTINUE
180       J1=MAX(J,J1)
          IF(J1.EQ.N) GOTO 200
          I=I+1
          GOTO 140
      ENDIF
C
C COMPUTE THE EXP OF THE CLUSTER
C
200   CONTINUE
      IF(J1.EQ.N) THEN
          DO 220 I=1,N
220       E(I,N)=B(I,N)*TAU
      ENDIF
      CALL CEXPRI(E(I1,I1),W,IP(I1),IDX(I1),B,OVFT,IERR,J1-I1+1,NE)
      DO 240 I=I1,J1
240   IDX(I)=I1
      I1=J1+1
      GOTO 120
260   CONTINUE
C
C RESTORE B
C
      DO 300 J=1,N/2
          K=N-J
          DO 280 I=1,J
280       B(I,J)=B(K+I,K)
300   CONTINUE
      DO 320 I=1,N
320   W(I)=TAU*B(I,I)
C
C FILL UP THE OFF-DIAGONAL BLOCK BY RECURRENCE
C
      DO 380 I=N,1,-1
      DO 360 J=I+1,N
          IF(IDX(I).NE.IDX(J).OR.AB(W(J)-W(I)).GT.G(J-I)) THEN
            Z=B(I,J)*(E(J,J)-E(I,I))
            DO 340 K=I+1,J-1
                Z=Z+B(I,K)*E(K,J)-E(I,K)*B(K,J)
340         CONTINUE
            Z=Z/(B(J,J)-B(I,I))
            E(I,J)=Z
          ENDIF
360   CONTINUE
380   CONTINUE
C
C CLEAR THE LOWER B AND E
C
      DO 440 J=1,N-1
          DO 400 I=J+1,N
400       E(I,J)=ZERO
          DO 420 I=J+1,N
420       B(I,J)=ZERO
440   CONTINUE
      RETURN
      END
```

CEXPRI
(Subprogram of CEXPHY)

1. PURPOSE.

The Fortran 77 subroutine CEXPRI computes the exponential of a complex triangular matrix E using a combination of the Newton polynomial method and Parlett's recurrence formula. The result overwrites E.

2. USAGE.

Calling Sequence.

SUBROUTINE CEXPRI(E,w,ip,idx,s,ovft,ierr,n,ne)

Parameters :

E	is a complex two-dimensional variable with row dimension nb and column dimension at least n. On input, E contains a complex triangular matrix of order n. On output, exp(E) overwrites E.
w	is a complex one-dimensional variable with dimension at least 5n. Work space.
ip	is an integer array of dimension n. Work space.
idx	is an integer array of dimension n. Work space.
s	is an one dimensional complex array of dimension at least n(n-2)/4. Work space.
ovft	is a real variable set equal to the overflow threshold.
ierr	is an integer variable indicating error condition. If the exponential of some eigenvalues overflow, ierr is set equal to -2, and the subroutine terminates.
n	is an input integer set equal to the column dimension of the matrix E.
ne	is an input integer set equal to the row dimension of E respectively.

Subprograms : CEXPNT, CINDEX, CMSWAP

3. DISCUSSION OF METHOD AND ALGORITHM.

Let z(1),z(2),...,z(n) denote the eigenvalues of E. CEXPRI first determines a grouping of z(i) by subroutine CINDEX. The criterion is : if the imaginary parts of z(i) and z(j) are differ by less than pi=3.14159... then z(i) and z(j) should belong to the same group. We use an integer array idx(i) to index z(i). Thus, if z(i) and z(j) should be together, we set idx(i)=idx(j).

After determining idx(i), we apply CMSWAP if necessary to have E's diagonal ordered according to idx(i) (via untary similarity transformations). Then on each group (diagonal block) subroutine CEXPNT is called to compute its exponential. Off diagonal elements are filled up by Parlett's recurrence (cf. [3]) (resutls overwrite E).

Finally, we undo the unitary similarity transformations on E.

There are approximately 42 executable Fortran statements in CEXPRI, 31 in subroutine CMSWAP, and 42 in subroutine CINDEX.

4. REFERENCES.

[1] G.H. Golub & C.F. VanLoan, *Matrix Computations,* Johns Hopkins University Press, 1983.

[2] A. McCurdy, K.C. Ng and B.N. Parlett, *Accurate Computation of Divided Differences of The Exponential Function,* Mathematics of Computation, Volume 43, Number 168, October 1984, pp501-528.

[3] B.N. Parlett, *A Recurrence Among the Elements of Functions of Triangular Matrices,* Linear Algebra Appl., 14(1976), pp.117-121.

5. PROGRAM LISTING.

```
      SUBROUTINE CEXPRI(E,W,IP,IDX,S,OVFT,IERR,N,NE)
      COMPLEX E(NE,N),S(N*(N-2)/4),W(5*N)
      INTEGER IP(N),IDX(N),N,NE,IERR
      REAL OVFT
C
C THIS SUBROUTINE COMPUTES THE EXPONENTIAL OF (COMPLEX TRIANGULAR) E BY
C CEXPNT AND PARLETT RECURRENCE. RESULT OVERWRITES E.
C CODE IN F77 BY K.C. NG, UC BERKELEY; REVISED ON 6/6/85.
C
C     F77  GENERIC FUNCTIONS      :  CONJG
C
C     REQUIRED SUBROUTINES        :  CINDEX, CEXPNT, CMSWAP
C
C GLOSSARY:
C     E      ON INPUT, AN UPPER TRIANGULAR (COMPLEX) MATRIX; ON OUTPUT,
C            THE EXPONENTIAL OF E
C     W,S    COMPLEX ARRAYS, WORK SPACE
C     IP,IDX INTEGER ARRAYS, WORK SPACE
C     OVFT   COMPUTER (REAL) OVERFLOW THRESHOLD
C     IERR   (INTEGER) ERROR STATUS: RETURN IERR= -2 IF SOME EIGENVALUE
C            IS TOO LARGE TO EXPONENTIATE.
C     N      DIMENSION OF MATRIX E
C     NE     LEADING DIMENSION OF ARRAY E
C
C INTERNAL VARIABLES:
C
      INTEGER I,J,K,L
      COMPLEX X
      DO 20 I=1,N
20    W(I)=E(I,I)
C
C FIND OUT THE NEW ORDERING FOR W(I)
C
      CALL CINDEX(N,W,W(N+1),W(2*N+1),IP,IDX)
C
C SWAP E(I,I) ACCORDING TO THE NEW ORDERING
C
      L=0
      DO 100 I=N,1,-1
           DO 40 J=I,1,-1
              IF(IDX(J).EQ.IP(I)) GOTO 60
40         CONTINUE
60         DO 80 K=J+1,I
              X=E(K-1,K)/(E(K,K)-E(K-1,K-1))
              L=L+1
              S(L)=CONJG(X)
              CALL CMSWAP(E,W,IDX,X,K,N,NE)
80         CONTINUE
           IP(I)=J+1
100   CONTINUE
      DO 110 I=1,N-1
      DO 110 J=I+1,N
110   E(J,I)=E(I,J)
C
C COMPUTE THE EXPONENTIAL OF THE DIAGONAL BLOCKS
C
      K=IDX(1)
      I=1
      DO 140 J=1,N
           IF(J.EQ.N) THEN
              CALL CEXPNT(E(I,I),W(I),OVFT,IERR,N-I+1,NE)
```

```
          ELSE IF(IDX(J).NE.K) THEN
            CALL CEXPNT(E(I,I),W(I),OVFT,IERR,J-I,NE)
            I=J
            K=IDX(I)
          ENDIF
          IF(IERR.EQ.-2) RETURN
140   CONTINUE
C
C COMPUTE THE OFF-DIAGONAL ELEMENTS BY PARLETT RECURRENCE
C
      DO 300 I=N,1,-1
      DO 300 J=I+1,N
          IF(IDX(I).NE.IDX(J)) THEN
            X=E(J,I)*(E(J,J)-E(I,I))
            DO 280 K=I+1,J-1
280         X=X+E(K,I)*E(K,J)-E(I,K)*E(J,K)
            X=X/(W(J)-W(I))
            E(I,J)=X
          ENDIF
300   CONTINUE
C
C SWAP BACK
C
      DO 320 I=1,N
      DO 320 K=I,IP(I),-1
          X=S(L)
          L=L-1
          CALL CMSWAP(E,W,IDX,X,-K,N,NE)
320   CONTINUE
      RETURN
      END


      SUBROUTINE CMSWAP(E,Z,IDX,X,K,N,NE)
      COMPLEX E(NE,N),Z(N),X
      INTEGER K,NE,N,IDX(N)
C
C THIS SUBROUTINE SWAPS THE K AND K-1 DIAGONAL ELEMENTS OF MATRIX E USING
C UNITARY C TRANSFORMATION (THE TRANSFORMATION IS ENCODED IN THE INPUT X).
C IT ALSO SWAPS THE CORRESPONDING ELEMENTS IN ARRAY Z AND ARRAY IDX.
C CODE IN F77 BY K.C. NG, UC BERKELEY; REVISED ON 5/18/85.
C
C      F77  GENERIC FUNCTIONS     :  ABS, CONJG, REAL, SQRT
C      F77  NON-GENERIC FUNCTION:  CMPLX
C
C GLOSSARY:
C      E      AN UPPER TRIANGULAR (COMPLEX) MATRIX
C      Z      COMPLEX ARRAY
C      IDX    INTEGER ARRAY
C      X      COMPLEX NUMBER CONTAINS THE TRANSFORMATION INFORMATION
C      K      INTEGER INDEX
C      N      DIMENSION OF MATRIX E
C      NE     LEADING DIMENSION OF ARRAY E
C
C INTERNAL VARIABLES
C
      COMPLEX U11,U12,U21,ZERO
      INTEGER I,J
      DATA ZERO/(0.0,0.0)/
C
C RECONSTRUCT THE UNITARY MATRIX FROM X.
C
      U21=CMPLX(1.0/SQRT(1.0+REAL(CONJG(X)*X)))
```

```
      U11=X*U21
      U12=U21
      IF(X.NE.ZERO) THEN
          IF(K.GT.0) THEN
            U12=-U11/CONJG(X)
          ELSE
            U21=-U11/CONJG(X)
          ENDIF
      ENDIF
      K=ABS(K)
C
C PERFORM E*U
C
      DO 30 I=1,K-2
          X        =E(I,K-1)*U11+E(I,K)*U21
          E(I,K)   =E(I,K-1)*U12+E(I,K)*U11
          E(I,K-1)=X
30    CONTINUE
C
C PERFORM UH*E
C
      U11=CONJG(U11)
      U21=CONJG(U21)
      U12=CONJG(U12)
      DO 40 J=K+1,N
          X        =E(K-1,J)*U11+E(K,J)*U21
          E(K,J)   =E(K-1,J)*U12+E(K,J)*U11
          E(K-1,J)=X
40    CONTINUE
C
C SWAP THE DIAGONAL OF E AND THE CORRESPONDING ELEMENTS IN Z AND IDX.
C
      I           =IDX(K)
      IDX(K)      =IDX(K-1)
      IDX(K-1)    =I
      X           =E(K,K)
      E(K,K)      =E(K-1,K-1)
      E(K-1,K-1)=X
      X           =Z(K)
      Z(K)        =Z(K-1)
      Z(K-1)      =X
      RETURN
      END



      SUBROUTINE CINDEX(N,W,ZI,IQ,IP,IDX)
      COMPLEX W(N)
      REAL ZI(N)
      INTEGER IP(N),IQ(N),IDX(N),N
C
C THIS SUBROUTINE ASSIGNS A GROUP NUMBER IDX(I) TO EACH OF THE IMAGINARY
C PARTS ZI(I) OF THE EIGENVALUES W(I). THE CRITERION IS : IF THE DISTANT
C BETWEEN ZI(I) AND ZI(J) ARE LESS THEN PI=3.14159 , THEN THEY HAVE
C THE SAME GROUP NUMBER.
C CODE IN F77 BY K.C. NG, UC BERKELEY, REVISED ON 5/18/85
C
C     F77 GENERIC FUNCTIONS      : ABS, ACOS, IMAG
C
C GLOSSARY:
C     W     INPUT COMPLEX ARRAY CONTAINING THE EIGENVALUES
C     ZI    REAL ARRAY CONTAINING THE IMAGINARY PARTS OF W
C     IDX   OUTPUT INTEGER ARRAY CONTAINING THE GROUP NUMBERS OF THE
C           CORRESPONDING ELEMENTS IN ZI
```

```
C       IP      OUTPUT INTEGER ARRAY CONTAINING THE NEW ORDERING OF ZI
C       IQ      INTEGER ARRAY, WORK SPACE
C       N       DIMENSION OF ARRAY W,ZI,IP,IQ
C
C.INTERNAL VARIABLES
C
        REAL PI,ZERO
        INTEGER I,J,K,L,IDIFF,ITEMP
        DATA ZERO/0.0/
        PI=2.0*ACOS(ZERO)
        DO 10 I=1,N
             IQ(I)=I+1
             ZI(I)=IMAG(W(I))
10      CONTINUE
        IP(1)=1
        IDX(1)=1
        L=1
        K=1
        I=1
C
C DETERMINE IDX(I)
C
20      IF(L.LT.N) THEN
30           IF(I.LE.N-L) THEN
               IF(ABS(ZI(IQ(I))-ZI(IP(K))).GE.PI) THEN
                    I=I+1
                    GOTO 30
               ELSE
                    ITEMP=IDX(IP(K))
               ENDIF
             ELSE
               I=1
               K=K+1
               IF(K.LE.L) GOTO 30
               ITEMP=IQ(I)
             ENDIF
             L=L+1
             IP(L)=IQ(I)
             IDX(IP(L))=ITEMP
             DO 40 J=I,N-L
40           IQ(J)=IQ(J+1)
             GOTO 20
        ENDIF
C
C PUT THE NEW ORDERING IN IP()
C
        IDIFF=0
        DO 120 J=2,N
        DO 120 I=1,J-1
             IF(IDX(J).GT.IDX(I)) THEN
                IDIFF=IDIFF-1
             ELSE IF(IDX(J).LT.IDX(I)) THEN
                IDIFF=IDIFF+1
             ENDIF
120     CONTINUE
        K=1
        DO 160 L=1,N
             J=L
             IF(IDIFF.GT.0) J=N+1-L
             DO 140 I=1,N
                IF(IDX(I).EQ.J) THEN
                     IP(K)=J
                     K=K+1
                ENDIF
```

```
140       CONTINUE
160    CONTINUE
       RETURN
       END
```

CEXPNT
(Subprogram of CEXPRI)

1. PURPOSE.

The Fortran 77 subroutine CEXPNT computes the exponential E of a complex triangular matrix using the Newton interpolating polynomial.

2. USAGE.

(A). Calling Sequence.

SUBROUTINE CEXPNT(B,w,ovft,ierr,n,nb)

Parameters :

B is a complex two-dimensional variable with row dimension nb and column dimension at least n. On input, B contains a complex triangular matrix of order n, with its diagonal elements ordered according to the real part of B(i,i). On output, it contains E .

w is a complex one-dimensional variable with dimension at least 5n. Work space.

ovft is the overflow threshold.

ierr is an integer variable indicating the error condition. If the exponential of some eigenvalues overflow, ierr is set equal to -2, and the subroutine terminates.

n is an input integer equal to the order of the matrix B.

nb is an input integer equal to the row dimension of B.

Subprogram : CDDEXP.

3. DISCUSSION OF METHOD AND ALGORITHM.

Before computing E=exp(B), we scale down the size of B so that the the maximum spread of the imaginary parts of B's eigenvalues is less then 5. Then exp(B) is computed by the Newton polynomial method from the bottom row to the top row.

Suppose that row j of exp(B) has been computed. To compute row j-1, we first evaluate the scaled divided differences $k!\Delta_i^k$exp for k=0,...,n-i where Δ_i^kexp denotes the k-th divided difference of exp at B's eigenvalues z(i),z(i+ 1),...,z(i+ k) (z(i) is the i-th diagonal element of B). This is done by calling subroutine CDDEXP. Then E(i,i),...,E(i,m) are computed using the Newton interpolating polynomial as follows.

Let B_i denote the principal submatrix of B in rows i through n, of B

$$B_i = \begin{pmatrix} z(i) & b_{i,i+1} & \cdot & \cdot & b_{i,n} \\ 0 & z(i+1) & \cdot & \cdot & b_{i+1,n} \\ 0 & 0 & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot \\ 0 & 0 & 0 & 0 & z(n) \end{pmatrix}.$$

The i-th row of exp(B) can be regarded as the first row of $\exp(B_i)$; which can be represented by

$$\exp(B_i) = \exp(z(i))\cdot I + \sum_{k=1}^{n-i} (k!\Delta_i^k \exp)\left(\frac{1}{k!}\prod_{j=i}^{k+i-1}(B_i - z(j)\cdot I)\right). \tag{1}$$

Only one complex vector is needed to accumulate the top row of the product matrix $\prod(B_i - z(j)\cdot I)$ and hence the top row of $\exp(B_i)$. The above is repeated until $i = 1$.

Finally any needed squarings are performed.

There are approximately 115 executable Fortran statements.

4. REFERENCES.

NONE

5. PROGRAM LISTING.

```
      SUBROUTINE CEXPNT(B,W,OVFT,IERR,N,NB)
      COMPLEX B(NB,N), W(5*N)
      INTEGER N,NB,IERR
      REAL OVFT
C
C THIS SUBROUTINE USES THE NEWTON POLYNOMIAL METHOD WITH SCALING AND
C SQUARING TO COMPUTE THE EXPONENTIAL OF B. RESULTS OVERWRITE B  THE
C OLD MATRIX B IS KEPT IN THE LOWER TRIANGULAR PART OF B.
C CODE IN F77 BY K.C. NG, UC BERKELEY; REVISED ON 6/17/85.
C
C      F77  GENERIC FUNCTIONS     :  ABS ,SIN, EXP, IMAG, MAX, MIN, REAL
C      F77  NON-GENERIC FUNCTION:  CMPLX
C
C      REQUIRED SUBROUTINES       :  CDDEXP
C
C      REQUIRED BLAS SUBROUTINES:  CAXPY, CCOPY, CSCAL, CSSCAL
C      REQUIRED BLAS FUNCTION    :  CDOTU
C
C GLOSSARY:
C      B       ON INPUT, AN UPPER TRIANGULAR (COMPLEX) MATRIX. ON OUTPUT,
C              THE EXPONENTIAL OF B, WITH THE ORIGINAL B STORED IN THE
C              LOWER TRIANGULAR PART (DIAGONALS ARE STORED IN W(1) TO W(N)).
C      W       COMPLEX ARRAY, WORK SPACE
C      OVFT    COMPUTER (REAL) OVERFLOW THRESHOLD
C      IERR    OUTPUT (INTEGER) ERROR STATUS: IF SOME EIGENVALUE OF B IS TOO
C              LARGE TO EXPONENTIATE, THE PROGRAM QUITS AND RETURNS IERR=-2.
C      N       DIMENSION OF MATRIX B
C      NB      LEADING DIMENSION OF ARRAY B
C
C INTERNAL VARIABLES:
C
      INTEGER I,J,K,L,M,N4,N2,KSQ
      COMPLEX X,Y,SHIFT,CDOTU
      REAL ZERO,DLIM,SCALE,ZIMAX,ZIMIN,A0,A1,A2,DIST
      REAL DMAX,TEMP,EMIN,PMAX,BMAX,AB,G
      DATA A0,A1,A2/-1.797804884,1.958414640,.02939024390/
      DATA DLIM,ZERO/5.0,0.0/
      AB(X)=ABS(REAL(X))+ABS(IMAG(X))
      G(I)=MAX(ZERO,A0+I*(A1+I*A2))
C
C IF REAL(B(N,N)) IS TOO LARGE, RETURN IERR=-2 AND TERMINATE.
C
      IF(REAL(B(N,N)).GT.LOG(OVFT)) THEN
          IERR=-2
          RETURN
      ENDIF
C
C IF N=1 OR N=2, USE SPECIAL FORMULA
C
      IF(N.LT.1) RETURN
      IF(N.EQ.1) THEN
          W(1)=B(1,1)
          B(1,1)=EXP(W(1))
          RETURN
      ENDIF
      IF(N.EQ.2) THEN
          W(1)=B(1,1)
          W(2)=B(2,2)
          B(1,1)=EXP(W(1))
          B(2,2)=EXP(W(2))
          B(2,1)=B(1,2)
```

```
            IF(W(1) EQ W(2)) THEN
              B(1,2)=B(1,2)*B(1,1)
            ELSE
              X=CMPLX(0 5)*(W(1)+W(2))
              Y=CMPLX(0.0,0 5)*(W(2)-W(1))
              B(1,2)=B(1,2)*EXP(X)*(SIN(Y)/Y)
            ENDIF
            RETURN
         ENDIF
C
C COMPUTE AND STORE THE EXPONENTIAL OF B IN THE LOWER TRIANGULAR B. THE
C DIAGONAL OF B IS STORED IN W(1), W(2), . . . ,W(N).
C
         N2=N+N
         N4=N2+N2
         ZIMAX=0.0
         ZIMIN=0.0
         DO 20 I=1,N
            W(I)=B(I,I)
            TEMP=IMAG(W(I))
            ZIMAX=MAX(ZIMAX,TEMP)
            ZIMIN=MIN(ZIMIN,TEMP)
20       CONTINUE
         CALL CCOPY(N,W,1,W(N4+1),1)
C
C SHIFT OR SCALE DOWN B TO GUARANTEE THE IMAGINARY PARTS ARE BOUNDED BY DLM
C
         SHIFT=CMPLX(0.0,0.0)
         SCALE=1
         KSQ=0
         IF(MAX(-ZIMIN,ZIMAX) GT DLIM) THEN
            IF(ZIMAX.LT.ZERO.OR.ZIMIN.GT.ZERO) SHIFT=
     X         CMPLX(0.0,(ZIMAX+ZIMIN)/2)
            DO 60 I=1,N
60          W(I)=W(I)-SHIFT
            DIST=ZIMAX-ZIMIN
80          IF(DIST.GT.DLIM) THEN
               SCALE=SCALE/2.0
               DIST=DIST/2.0
               KSQ=KSQ+1
               GOTO 80
            ENDIF
            IF(KSQ.GT.0) THEN
               DO 120 J=1,N
                  CALL CSSCAL(J-1,SCALE,B(1,J),1)
120            CONTINUE
              CALL CSSCAL(N,SCALE,W,1)
            ENDIF
         ENDIF
C
C  COMPUTE THE EXP OF B FROM BOTTOM TO TOP (RESULTS STORE IN THE LOWER B)
C
         DO 320 I=N,1,-1
            DO 140  J=N,I,-1
              DIST=AB(W(I)-W(J))
              IF(DIST LE.G(J-I)) GOTO 160
140         CONTINUE
C
C  COMPUTE THE SCALE DIVIDED DIFFERENCES AT W(I),     ,W(N). RESULTS STORE
C  IN W(N+I), . . . ,W(2N)
C
160         CALL CDDEXP(W(I),W(N+I),W(N2+1),W(N2+N+1),J-I+1,OVFT)
            DO 180 K=J+1,N
180         W(K+N)=CMPLX(REAL(K-I))*(W(K+N)-W(K+N-1))/(W(K)-W(I))
```

```
            DO 200 K=I,J
200         W(K)=W(N4+K)-SHIFT
            CALL CSSCAL(J-I+1,SCALE,W(I),1)
            B(I,I)=EXP(W(I))
            IF(I.EQ.N) GOTO 320
            DMAX=0.0
            DO 220 K=N+I,N2
220         DMAX=MAX(DMAX,AB(W(K)))
C
C  COMPUTE E(I,I+1),....,E(I,N) USING NEWTON POLYNOMIAL METHOD
C
            Y=W(N+I+1)
            DO 240 K=I+1,N
              W(K+N2)=B(I,K)
              B(K,I)=Y*W(K+N2)
240         CONTINUE
            DO 300 M=I+2,N
              Y=W(M+N)
              EMIN=OVFT
              BMAX=AB(W(M-1))
              PMAX=0.0
              DO 280 K=N,M,-1
                   X=W(K+N2)*(W(K)-W(M-1))+
     X               CDOTU(K-M+1,W(N2+M-1),1,B(M-1,K),1)
                   TEMP=M-I
                   X=X/TEMP
                   W(K+N2)=X
                   B(K,I)=B(K,I)+X*Y
                   PMAX=MAX(PMAX,AB(X))
                   EMIN=MIN(EMIN,AB(B(K,I)))
                   BMAX=MAX(BMAX,AB(W(K)))
                   DO 260 L=M-1,K-1
260                BMAX=MAX(BMAX,AB(B(L,K)))
280           CONTINUE
              TEMP=-1.0
              IF(BMAX.LT.1.0) TEMP=EMIN+PMAX*DMAX*BMAX
C
C TEST FOR AN EARLY CONVERGENCE
C
              IF(TEMP.EQ.EMIN) GOTO 320
              IF(PMAX.EQ.ZERO) GOTO 320
300         CONTINUE
320     CONTINUE
C
C ***** END OF COMPUTING THE EXPONENTIAL OF THE SCALED B *****
C
C TRANSPOSE B AND RESTORE THE ORIGINAL B IN THE LOWER PART
C
        SCALE=1.0/SCALE
        DO 330 I=1,N-1
        DO 330 J=I+1,N
            X=B(I,J)
            B(I,J)=B(J,I)
            B(J,I)=X*SCALE
330     CONTINUE
C
C SQUARE B KSQ TIMES
C
        DO 420 K=1,KSQ
            DO 340 J=1,N
340         W(J)=W(J)+W(J)
            DO 400 J=N,1,-1
              CALL CCOPY(J,B(1,J),1,W(N+1),1)
              CALL CSCAL(J-1,W(N+J),B(1,J),1)
```

```
              DO 380 I=1,J-1
                 CALL CAXPY(1,W(N+1),B(1,I),1,B(1,J),1)
380           CONTINUE
              B(J,J)=EXP(W(J))
400        CONTINUE
420     CONTINUE
C
C SHIFT BACK EXP(B) AND RESTORE THE DIAGONAL OF B IN W(1) TO W(N)
C
        X=EXP(SHIFT)
        CALL CCOPY(N,W(N4+1),1,W,1)
        DO 460 J=1,N
           CALL CSCAL(J,X,B(1,J),1)
460     CONTINUE
        RETURN
        END
```

CDDEXP
(Subprogram of CEXPNT)

1. PURPOSE

The Fortran 77 subroutine CDDEXP computes the scaled divided differences at data z(1), z(2),...,z(n). They are used in computing the matrix exponential by Newton polynomial method. See CEXPNT.

2. USAGE

(A). Calling Sequence.

SUBROUTINE CDDEXP(z,d,r,s,n,ovft)

Parameters:

z	is a complex one-dimensional input variable with dimension n. It contains the complex data at which the scaled divided differences are computed.
d	is a complex one-dimensional output variable with dimension n. On output, d contains the scaled divided differences at z(1), z(2),..., z(n).
r,s	are two complex one-dimensional variables of order n. Work space.
n	is an input integer equal to the dimension of arrays z,d,r and s.
ovft	is the biggest positive real number, overflow threshold.

3. DISCUSSION OF METHOD AND ALGORITHM

The scaled exponential divided differences at z(1),...,z(n) (see [1]) is the first row of the exponential of a bi-diagonal matrix G_z ={ G_z(i,i)=z(i), i=1,...,n, G_z(i,i+1)=i, i=1,...,n-1}. Here we apply a variation of the scaling and squaring method (See [2]) to compute exp(G_z). We first scale down the diameter of {z(i)} (by multipling 2^{-k}) to about 0.7. Then we use the Taylor serie to compute exp(G_z). Finally we scale and square exp(G_z) k time. Because the the special struction of G_z, our program keeps down storage needs to a minimum.

For high accuracy, the data z(1),...,z(n) must be ordered by their real parts: Re(z(1)) <= Re(z(2)) <= ... <= Re(z(n)).

There are approximately 75 executable Fortran statements.

4. REFERENCES

[1] A. McCurdy, K.C. Ng and B.N. Parlett, *Accurate Computation of Divided Differences of The Exponential Function,* Mathematics of Computation,

Volume 43, Number 168, October 1984, pp501-528.

[2] C. Moler and C. van Loan, *Nineteen Dubious Ways to Compute The Exponential of a Matrix,* SIAM Review, Vol. 20, No. 4, October 1978, p801-836.

5. PROGRAM LISTING.

```
      SUBROUTINE CDDEXP(Z,D,S,R,N,OVFT)
      COMPLEX Z(N),D(N),S(N),R(N)
      REAL OVFT
      INTEGER N
C
C THIS SUBROUTINE COMPUTES THE (COMPLEX) SCALED EXPONENTIAL DIVIDED
C DIFFERENCES AT Z(1),...,Z(N) BY SCALING AND SQUARING.
C WE ASSUME RE(Z(1)) <= ... <= RE(Z(N)).
C CODE IN F77 BY KWOK-CHOI NG, UC BERKELEY; REVISED ON 5/18/85.
C
C     F77 GENERIC FUNCTIONS    : ABS, EXP, LOG, MAX, REAL, IMAG
C     F77 NON-GENERIC FUNCTION: CMPLX
C
C     REQUIRED BLAS SUBROUTINES: CCOPY, CSSCAL, CSCAL
C
C GLOSSARY:
C     Z       INPUT COMPLEX ABSCISSAE
C     D       OUTPUT COMPLEX SCALED EXPONENTIAL DIVIDED DIFFERENCES
C     S,R     COMPLEX WORKING SPACE
C     OVFT    COMPUTER (REAL) OVERFLOW THRESHOLD
C     N       DIMENSION OF ARRAY Z,D,S,R
C
C INTERNAL VARIABLES
C
      COMPLEX SHIFT,X,Y,C
      INTEGER I,J,K
      REAL T,T1,T2,TE,SCALE,RADIU,RLIM
      DATA RLIM/0.7E0/
C
C SHIFT AND SCALE DOWN THE ABSCISSAE AND DETERMINE THE NUMBER OF SQUARING
C
      SHIFT = CMPLX(0.0)
      DO 10 I=1,N
10    SHIFT=SHIFT+Z(I)
      T=N
      SHIFT=SHIFT/T
C
C IF EXP(Z(N)-SHIFT) OVERFLOWS, THEN SET RE(SHIFT)=RE(Z(N)).
C
      IF(REAL(Z(N)-SHIFT).GT.LOG(OVFT)) SHIFT=
     XCMPLX(REAL(Z(N)),IMAG(SHIFT))
      DO 20 I=1,N
20    Z(I)=Z(I)-SHIFT
      RADIU=0
      DO 30 I=1,N
30    RADIU=MAX(RADIU,ABS(Z(I)))
      IF (RADIU.LE.RLIM) THEN
         K=0
      ELSE
         K=1+(LOG(RADIU)+0.3566749)/0.6931472
      ENDIF
C
C SCALE DOWN Z: Z := Z/2**K SUCH THAT |Z|<0.7 ; INITIALIZE S AND D
C
      T=2**K
      SCALE=1.0/T
      RADIU=RADIU*SCALE
      DO 40 I=1,N
40    D(I)=CMPLX(1.0)
      DO 50 I=1,N
50    R(I)=CMPLX(0.0)
```

```
      CALL CCOPY(N,D,1,S,1)
      CALL CSSCAL(N,SCALE,Z,1)
C
C SUMMING THE TAYLOR SERIES : WHILE J=0,1,.. UNTIL CONVERGE DO ...
C
      J=0
      T1=EXP(-RADIU)
      TE=1
60    J=J+1
      T=J
      TE=TE*RADIU/T
      T2=T1+TE
      IF(T1.NE.T2) THEN
          X=Z(1)*S(1)
          S(1)=X/T
          DO 70 I=2,N
            T=I-1
            C=Z(I)*S(I)+S(I-1)*T
            T=J+T
            S(I)=C/T
            D(I)=D(I)+S(I)
70        CONTINUE
          GOTO 60
      ENDIF
      D(1)=EXP(Z(1))
C
C NOW SQUARE
C
100   CONTINUE
      IF(K.EQ.0) THEN
          X=EXP(SHIFT)
          DO 110 I=1,N
110       Z(I)=Z(I)+SHIFT
          CALL CSCAL(N,X,D,1)
      ELSE
          SCALE=1.0
          S(1)=D(2)
          DO 140 I=2,N
            X=S(1)
            S(1)=D(1)
            S(I)=EXP(Z(I))
            R(I)=(D(1)+S(I))*D(I)
            DO 130 J=2,I-1
                Y=S(J)
                C=S(J-1)*(Z(I)-Z(J-1))
                T=I-1
                C=C+X*T
                T=J-1
                S(J)=C/T
                R(I)=R(I)+S(J)*D(J)
                X=Y
130         CONTINUE
            SCALE=SCALE/2.0
            R(I)=R(I)*SCALE
140       CONTINUE
          DO 150 I=1,N
150       Z(I)=Z(I)+Z(I)
          CALL CCOPY(N,R,1,D,1)
          D(1)=EXP(Z(1))
          K=K-1
          GOTO 100
      ENDIF
      RETURN
      END
```

CFMULV

1. PURPOSE.

The Fortran 77 subroutine CFMULV computes the matrix product from right to left:

$$P \cdot F \cdot P^{*} \cdot V.$$

Here F is complex triangular.

2. USAGE.

(A). Calling Sequence.

SUBROUTINE CFMULV(F,P,V,w,n,nf,np,nv,k)

Parameters :

- F — is an input complex two-dimensional variable with row dimension nf and column dimension at least n. F contains a complex triangular matrix of order n.
- P — is an input complex two-dimensional variable with row dimension np and column dimension at least n. P contains an unitary matrix.
- V — is a complex two-dimensional variable with row dimension nv and column dimension at least k. On output, V is overwritten by the product $PFP^{*}V$.
- w — is a complex one-dimensional variable with dimension at least n. Work space.
- n — is a input integer equal to the order of the matrix S.
- nf,np — are input integers equal to the row dimension of F and P respectively.
- nv,k — are input integers equal to the row and column dimension of V respectively.

3. DISCUSSION OF METHOD AND ALGORITHM.

There are approximately 12 executable Fortran statements.

4. REFERENCES.

NONE

5. PROGRAM LISTING.

```
      SUBROUTINE CFMULV(F,P,V,W,N,NF,NP,NV,K)
      COMPLEX F(NF,N),P(NP,N),V(NV,K),W(N)
      INTEGER N,NF,NP,NV,K
C
C THIS SUBROUTINE EVALUATES V := P•(F•(PT•V))
C CODE IN F77 BY K.C. NG, UC BERKELEY; REVISED ON 6/8/85.
C
C     REQUIRED BLAS SUBROUTINE :  CAXPY
C     REQUIRED BLAS FUNCTION   :  CDOTC
C
C GLOSSARY:
C     F       COMPLEX TRIANGULAR MATRIX
C     P       UNITARY MATRIX
C     V       N BY K MATRIX: ON OUTPUT, IT IS REPLACED BY P•(F•(PT•V))
C     W       COMPLEX ARRAY, WORK SPACE
C     N       DIMENSION OF MATRICES F,P,V
C     NF      LEADING DIMENSION OF ARRAY F
C     NP      LEADING DIMENSION OF ARRAY P
C     NV      LEADING DIMENSION OF ARRAY V
C
C INTERNAL VARIABLES:
C
      INTEGER I,J
      COMPLEX CDOTC, T, ZERO
      DATA ZERO/(0E0,0E0)/
C
C DO ONE COLUMN AT A TIME
C
      DO 100 J=1,K
C
C FORM  W := F•(PT•V(.,J))
C
          DO 20 I=1,N
20        W(I)=ZERO
          DO 40 I=1,N
            T=CDOTC(N,P(1,I),1,V(1,J),1)
            CALL CAXPY(I,T,F(1,I),1,W,1)
40        CONTINUE
C
C FORM V(.,J) = P • W
C
          DO 60 I=1,N
60        V(I,J)=ZERO
          DO 80 I=1,N
80        CALL CAXPY(N,W(I),P(1,I),1,V(1,J),1)
100   CONTINUE
      RETURN
      END
```

BLAS_C

1. PURPOSE.

The Basic Linear Algebra Subroutines (Complex) compute the basic matrix-vector operation. See [1].

2. USAGE.

(A). Calling Sequence.

```
SUBROUTINE   CAXPY (N,CA,CX,INCX,CY,INCY)
SUBROUTINE   CCOPY (N,CX,INCX,CY,INCY)
SUBROUTINE   CSSCAL(N,SA,CX,INCX)
SUBROUTINE   CSCAL (N,CA,CX,INCX)
COMPLEX FUNCTION CDOTU   (N,CX,INCX,CY,INCY)
COMPLEX FUNCTION CDOTC   (N,CX,INCX,CY,INCY)
```

Parameters :

See listing.

3. DISCUSSION OF METHOD AND ALGORITHM.

See [1].

There are approximately 75 executable Fortran statements.

4. REFERENCES.

[1]. J.J. Dongarra, C.B. Moler, J.R. Bunch, and G.W. Stewart, *LINPACK users' guide,* SIAM, Philadelphia/1979.

5. PROGRAM LISTING.

```
C****************************************************************
C    LINPACK BLAS FUNCTIONS AND SUBROUTINES :                    *
C                  CDOTC, CDOTU                                  *
C                  CAXPY, CCOPY, CSSCAL, CSCAL                   *
C****************************************************************

      SUBROUTINE CAXPY(N,CA,CX,INCX,CY,INCY)
C
C     CONSTANT TIMES A VECTOR PLUS A VECTOR.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      COMPLEX CX(1),CY(1),CA
      INTEGER I,INCX,INCY,IX,IY,N
C
      IF(N.LE.0)RETURN
      IF (ABS(REAL(CA)) + ABS(AIMAG(CA)) .EQ. 0.0 ) RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C        CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C          NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        CY(IY) = CY(IY) + CA*CX(IX)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
   20 DO 30 I = 1,N
        CY(I) = CY(I) + CA*CX(I)
   30 CONTINUE
      RETURN
      END

      SUBROUTINE  CCOPY(N,CX,INCX,CY,INCY)
C
C     COPIES A VECTOR, X, TO A VECTOR, Y.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      COMPLEX CX(1),CY(1)
      INTEGER I,INCX,INCY,IX,IY,N
C
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C        CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C          NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
```

```
          CY(IY) = CX(IX)
          IX = IX + INCX
          IY = IY + INCY
   10 CONTINUE
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
   20 DO 30 I = 1,N
        CY(I) = CX(I)
   30 CONTINUE
      RETURN
      END


      COMPLEX FUNCTION CDOTU(N,CX,INCX,CY,INCY)
C
C     FORMS THE DOT PRODUCT OF TWO VECTORS.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      COMPLEX CX(1),CY(1),CTEMP
      INTEGER I,INCX,INCY,IX,IY,N
C
      CTEMP = (0.0,0.0)
      CDOTU = (0.0,0.0)
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C        CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C          NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        CTEMP = CTEMP + CX(IX)*CY(IY)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      CDOTU = CTEMP
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
   20 DO 30 I = 1,N
        CTEMP = CTEMP + CX(I)*CY(I)
   30 CONTINUE
      CDOTU = CTEMP
      RETURN
      END


      COMPLEX FUNCTION CDOTC(N,CX,INCX,CY,INCY)
C
C     FORMS THE DOT PRODUCT OF TWO VECTORS, CONJUGATING THE FIRST
C     VECTOR.
C     JACK DONGARRA, LINPACK,  3/11/78.
C
      COMPLEX CX(1),CY(1),CTEMP
      INTEGER I,INCX,INCY,IX,IY,N
C
      CTEMP = (0.0,0.0)
      CDOTC = (0.0,0.0)
```

```
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C        CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C          NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        CTEMP = CTEMP + CONJG(CX(IX))*CY(IY)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      CDOTC = CTEMP
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
   20 DO 30 I = 1,N
        CTEMP = CTEMP + CONJG(CX(I))*CY(I)
   30 CONTINUE
      CDOTC = CTEMP
      RETURN
      END

      SUBROUTINE  CSSCAL(N,SA,CX,INCX)
C
C     SCALES A COMPLEX VECTOR BY A REAL CONSTANT.
C     JACK DONGARRA, LINPACK, 3/11/78.
C     MODIFIED FOR F77 BY K.C. NG, 4/14/85.
C
      COMPLEX CX(1)
      REAL SA
      INTEGER I,INCX,N,NINCX
C
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GO TO 20
C
C        CODE FOR INCREMENT NOT EQUAL TO 1
C
      NINCX = N*INCX
      DO 10 I = 1,NINCX,INCX
10    CX(I) = CX(I)*SA
      RETURN
C
C        CODE FOR INCREMENT EQUAL TO 1
C
20    DO 30 I = 1,N
30    CX(I) = CX(I)*SA
      RETURN
      END

      SUBROUTINE  CSCAL(N,CA,CX,INCX)
C
C     SCALES A VECTOR BY A CONSTANT.
C     JACK DONGARRA, LINPACK,  3/11/78.
C
      COMPLEX CA,CX(1)
      INTEGER I,INCX,N,NINCX
C
```

```
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GO TO 20
C
C        CODE FOR INCREMENT NOT EQUAL TO 1
C
      NINCX = N*INCX
      DO 10 I = 1,NINCX,INCX
        CX(I) = CA*CX(I)
   10 CONTINUE
      RETURN
C
C        CODE FOR INCREMENT EQUAL TO 1
C
   20 DO 30 I = 1,N
        CX(I) = CA*CX(I)
   30 CONTINUE
      RETURN
      END
```

Legal Notice

END

FILMED

11-85

DTIC